Nama=Daffa Dermawan

NIM=1124160177

-------------------------------------------------------------------------------------------------------------

APP Finance\_Track

SourceCode\_FinanceTrack(App)

import tkinter as tk

import customtkinter as ctk

from PIL import Image

import random

from tkcalendar import DateEntry

import time

from tkinter import messagebox

import sqlite3

Library yang digunakan:

Saya menggunakan customtkinter sebagai base dari GUI yang saya terapkan pada program yang saya buat karena tampilannya lebih modern dibanding tkinter, tetapi tetap import tkinter karena untuk melengkapi kekurangan dari widget yang disediakan customtkinter, sqlite3 digunakan sebagai database yang digunakan sebagai tempat menyimpan akun dan catatan dari user.

Untuk library sisanya, saya gunakan sebagai pelengkap untuk fitur yang ada dalam aplikasi, seperti random berfungsi untuk mengambil elemen secara acak dari data yang disediakan. Lalu, time untuk menyediakan fungsi terkait waktu. Tkcalendar sebagai widget kalender, dan PIL saya gunakan untuk memuat gambar dari file untuk dihadirkan dalam program.

#Connect to the database

conn = sqlite3.connect('fintrack.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS account (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT NOT NULL UNIQUE,

password TEXT NOT NULL)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS financial\_record (

id INTEGER PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date DATE NOT NULL,

description TEXT,

amount REAL NOT NULL,

type TEXT CHECK(type IN ('income', 'expense')) NOT NULL,

FOREIGN KEY (account\_id) REFERENCES account(id) ON DELETE CASCADE)

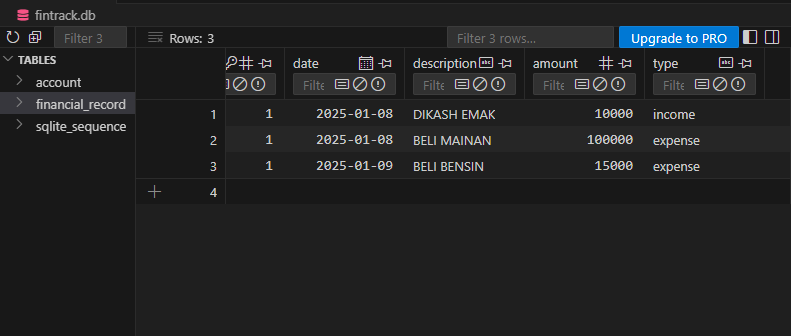
''')

conn.commit()

conn.close()

ScriptDatabase:

Variabel conn di gunakan untuk menghubungkan program pada database (jika tidak ada maka akan dibuat), lalu cursor digunakan untuk membuat objek dalam menjalankan perintah SQL di database. dan untuk selanjutnya cursor menjalankan perintah untuk membuat tabel account (terdiri dari; id, username, password) dan tabel financial\_record (terdiri dari; id, account\_id,date, description, amount, type (terdapat FOREIGN KEY yang memberikan relasi pada tabel account melalui kolom “account\_id”)), dan terakhir terdapat commit() dan close() yang merupakan perintah untuk menyimpan perubahan yang telah dilakukan dan menutup database untuk menghidari rusaknya database akibat program yang tiba-tiba error.



class App:

def \_\_init\_\_(self):

self.window = ctk.CTk()

self.window.title("Demo Aplikasi")

self.window.geometry("1024x576")

self.window.configure(bg="#64644e")

self.window.resizable(False, False)

ctk.set\_appearance\_mode("system")

ctk.set\_default\_color\_theme("green")

self.quotes = [

"Save first, spend later. - Unknown",

"Wealth is the ability to fully experience life. - Thoreau",

"Time is money. - Benjamin Franklin",

"Risk comes from ignorance. - Warren Buffett",

"A penny saved is a penny earned. - Benjamin Franklin",

"Small steps lead to big gains. - Unknown",

"Invest in yourself. - Unknown",

"Plan your work, work your plan. - Napoleon Hill",

"Less debt, more freedom. - Unknown",

"The best investment is in yourself. - Warren Buffett",

"Money grows when managed well. - Unknown",

"Every expense tells a story. - Unknown",

"Simplicity is the key to success. - Unknown",

"Budgeting is the first step to wealth. - Unknown",

"Opportunity favors the prepared. - Unknown",

"Earn more, spend less. - Unknown",

"Patience builds wealth. - Unknown",

"Control your money, or it controls you. - Unknown",

"Your habits define your future. - Unknown",

"A strong foundation builds lasting wealth. - Unknown"]

self.po\_image = Image.open("Finance (3).png")

self.po = ctk.CTkImage(self.po\_image, size=(150,120))

self.image = Image.open("Dashboard (1).png")

self.photo = ctk.CTkImage(self.image, size=(595,66))

self.logo\_image = Image.open('F (4).png')

self.logo =ctk.CTkImage(self.logo\_image, size=(320,576))

self.hello\_image = Image.open('81BFDA.png')

self.hello = ctk.CTkImage(self.hello\_image, size=(598,170))

self.income\_image =Image.open('1.png')

self.income = ctk.CTkImage(self.income\_image, size=(296,140))

self.expense\_image =Image.open('INCOME.png')

self.expense = ctk.CTkImage(self.expense\_image, size=(296,140))

self.logout\_image = Image.open('Desain tanpa judul (1).png')

self.logout = ctk.CTkImage(self.logout\_image,size=(50,50))

self.font1 = ("arial", 14, 'bold')

self.font2 = ("arial", 49, 'bold')

self.font3 = ("arial", 44,"bold")

self.font4 = ("arial", 28, "bold")

self.is\_signup\_mode = False

self.current\_user\_id = None

self.logged\_in\_username = None

self.login\_page()

Disini saya membuat class utama bernama App dimana tujuan pembuatannya adalah untuk membangun sebuah aplikasi GUI (menggunakan customtkinter), didalam fungsi \_\_init\_\_ saya memakainya untuk:

1. Membuat Jendela Utama Aplikasi

* Dimulai dari “self.window = ctk.CTk()” untuk membuat jendela utama dari aplikasi, lalu kode-kode selanjutnya berfungsi untuk mengatur nama jendela, ukuran jendela, warna background, lalu pembatas ukuran jendela agar tidak dapat diubah, juga pengatur tema untuk menyesuaikan tampilan aplikasi.

1. Menyiapkan data, gambar, format font, dan inisialisasi variabel

* Data yang dimaksud adalah kumpulan quote yang nantinya akan ditampilkan secara random pada “main\_page()”
* Semua gambar yang dibutuhkan aplikasi pun berada disini untuk dapat dipanggil sesuai kebutuhan tiap page
* Format font yang digunakan sebagai atribut dalam widget
* Dan inisialisasi variabel yang cukup krusial dalam jalannya aplikasi. (contoh mode signup/login atau informasi pengguna yang login.)
* Singkatnya untuk **Mempermudah akses atribut di metode lain,** dimana semua atribut yang dimulai dengan self bisa digunakan oleh metode lain dalam class, seperti **login\_page** atau **main\_page**.

1. Dan terakhir fungsi init ini digunakan sebagai tempat untuk menampilkan halaman, dan dikarenakan halaman utama yang ingin ditampilkan disini adalah halaman login maka yang dipanggil pertama adalah metode “login\_page()” nya

def login\_page(self):

def submit\_click(event=None):

conn = sqlite3.connect('fintrack.db')

cursor = conn.cursor()

username = entry\_username.get()

password = entry\_password.get()

self.logged\_in\_username = username

if not username or not password:

messagebox.showerror("Error", "Username dan password tidak boleh kosong!")

return

elif self.is\_signup\_mode:

try:

cursor.execute("INSERT INTO account (username, password) VALUES (?, ?)", (username, password))

conn.commit()

conn.close()

messagebox.showinfo("Registration Successful." ,"Account successfully created!")

switch\_to\_login()

entry\_username.delete(0, 'end')

entry\_password.delete(0, 'end')

except sqlite3.IntegrityError:

messagebox.showerror("Error","Username is already taken!")

else:

cursor.execute("SELECT id, username FROM account WHERE username = ?", (username,))

user = cursor.fetchone()

if user:

cursor.execute('''SELECT id, username FROM account WHERE username = ? AND password = ?''', (username, password))

user\_with\_pw=cursor.fetchone()

if user\_with\_pw:

self.current\_user\_id = user\_with\_pw[0]

messagebox.showinfo("Success", "LOGIN SUCCESSFUL!")

conn.close

for widget in self.window.winfo\_children():

widget.destroy()

self.main\_page()

else:

messagebox.showerror("Login Failed", "PASSWORD INCORRECT!")

conn.close()

else:

messagebox.showerror("Error", "ACCOUNT IS NOT EXIST!!")

def switch\_to\_signup():

self.is\_signup\_mode = True

entry\_username.focus()

entry\_username.delete(0, 'end')

entry\_password.delete(0, 'end')

print(self.is\_signup\_mode)

label\_header.configure(text="SIGN UP")

button\_submit.configure(text="Sign up", font= self.font1)

button\_createaccount.configure(text="Register account",command=switch\_to\_login, width=190, height=38)

def switch\_to\_login():

self.is\_signup\_mode = False

entry\_username.focus()

entry\_username.delete(0, 'end')

entry\_password.delete(0, 'end')

print(self.is\_signup\_mode)

label\_header.configure(text="LOGIN")

button\_submit.configure(text="Login", font=self.font1)

button\_createaccount.configure(text="Already have an account",command=switch\_to\_signup, width=190, height=38)

def focus\_password(event):

entry\_password.focus()

# Halaman login

frame1 = tk.Frame(self.window, bg="#8EAC50")

frame2 = ctk.CTkFrame(frame1, fg\_color="#002250", border\_width=4, border\_color="#002250")

frame1.place(x=0, y=0, relwidth=1, relheight=1)

frame2.place(x=15, y=15, relheight=0.95)

label\_logo = ctk.CTkLabel(frame2, text='', image=self.logo)

label\_logo.pack(pady=5)

label\_header = ctk.CTkLabel(frame1, text="LOGIN", font=self.font2, text\_color="#ffde59")

entry\_username = ctk.CTkEntry(frame1, width=350, height=45, placeholder\_text="Username",fg\_color="#E4F1AC",border\_color="#002250")

entry\_password = ctk.CTkEntry(frame1, width=350, height=45, placeholder\_text="Password", fg\_color="#E4F1AC", border\_color="#002250",show="\*")

button\_submit = ctk.CTkButton(frame1, text='Login', width=190, height=38, command=submit\_click,text\_color="#E7FBB4", fg\_color="#130170")

label\_pembatas = ctk.CTkLabel(frame1, text="-----------------------------", font=("arial", 35, "bold"), text\_color="#ffde59")

button\_createaccount = ctk.CTkButton(frame1, text="Don't have an account?", width=190, height=38, text\_color="#E7FBB4", cursor='hand2',command=switch\_to\_signup, fg\_color="#130170")

label\_header.place(x=680, y=120, anchor='center')

entry\_username.place(x=680, y=270, anchor=tk.CENTER)

entry\_username.bind("<Return>", focus\_password)

entry\_password.place(x=680, y=330, anchor=tk.CENTER)

entry\_password.bind("<Return>", submit\_click)

button\_submit.place(x=588, y=380)

label\_pembatas.place(x=678, y=450, anchor='center')

button\_createaccount.place(x=588, y=480)

Dalam metode login\_page(self), selain berisi GUI tampilan dari halaman login, tetapi juga terdapat beberapa fungsi yang didefinisikan untuk menangani berbagai aspek dari halaman login, seperti pengelolaan form login, proses registrasi, dan pengalihan antar tampilan.

Berikut adalah penjelasan mengenai fungsi-fungsi yang ada di dalamnya:

1. **submit\_click(event=None)**

* Ini adalah fungsi utama yang dipanggil ketika pengguna menekan tombol Login atau Sign up.
* Mengambil data username dan password yang dimasukkan pengguna.
* Melakukan validasi untuk memastikan username dan password tidak kosong.
* Jika dalam mode signup:
  + Mencoba untuk mendaftarkan pengguna baru dengan memasukkan username dan password ke dalam database.
  + Menampilkan pesan sukses atau gagal berdasarkan hasil registrasi.
* **Jika dalam mode login:**
  + Memeriksa apakah kombinasi username dan password sesuai dengan data yang ada di database.
  + Jika login berhasil, halaman login akan diganti dengan halaman utama aplikasi.
  + Jika login gagal, menampilkan pesan kesalahan.

1. **Switch\_to\_signup():**

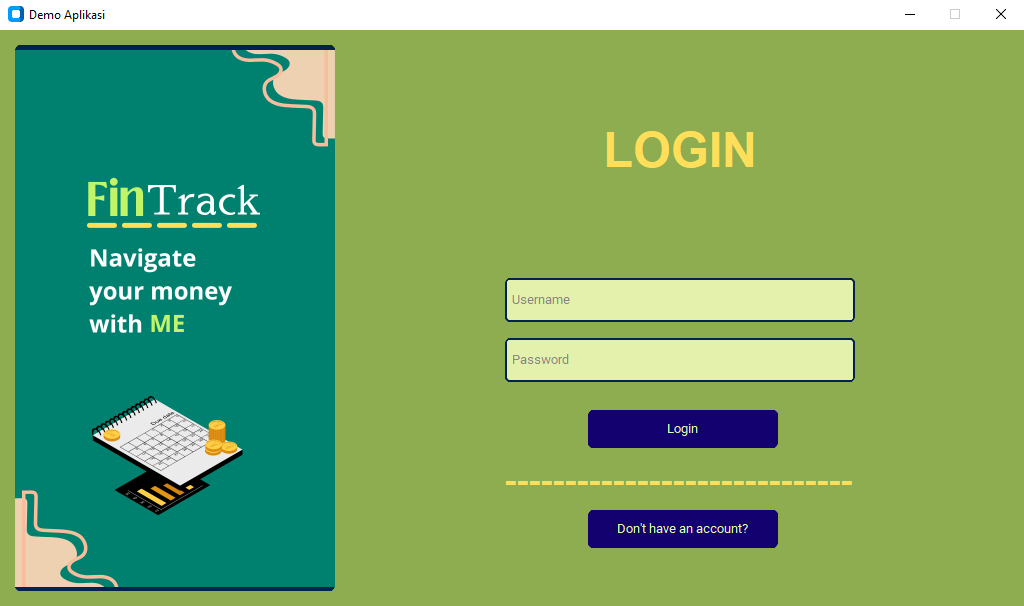
* Fungsi ini dipanggil ketika pengguna ingin berpindah dari login ke signup.
* Mengubah mode aplikasi ke signup (pembentukan akun baru).
* Menyesuaikan teks dan tombol di halaman agar sesuai dengan tampilan registrasi.
* Menghapus teks yang ada di form username dan password, serta memfokuskan form username untuk memudahkan input.

1. **Switch\_to\_login():**

* Fungsi ini dipanggil ketika pengguna ingin berpindah dari signup kembali ke login.
* Mengubah mode aplikasi kembali ke login.
* Menyesuaikan teks dan tombol di halaman agar sesuai dengan tampilan login.
* Menghapus teks yang ada di form username dan password, serta memfokuskan form username untuk memudahkan input.

1. **Focus\_password(event):**

* Fungsi ini dipanggil ketika pengguna menekan tombol Enter setelah memasukkan username.

Dan setelahnya terdapat elemen-elemen GUI yang Mengatur tampilan dan layout elemen-elemen seperti frame, label, entry, dan button agar sesuai dengan tampilan login atau signup.

def main\_page(self):

def logout():

self.current\_user\_id = None

self.logged\_in\_username = None

for widget in self.window.winfo\_children():

widget.destroy()

self.login\_page()

messagebox.showinfo("Logout", "You have been logged out!")

def search\_transactions():

conn = sqlite3.connect('fintrack.db')

cursor = conn.cursor()

date = combobox\_CRUD\_date.get()

if not date:

messagebox.showerror("Error", "Tanggal tidak boleh kosong.")

return

query = '''

SELECT date, description, amount, type

FROM financial\_record

WHERE account\_id = ?

AND date = ?

'''

try:

cursor.execute(query, (self.current\_user\_id, date))

results = cursor.fetchall()

print(f"Results: {results}")

for item in self.tree.get\_children():

self.tree.delete(item)

if results:

for row in results:

self.tree.insert("", "end", values=row)

else:

messagebox.showinfo("Info", "Tidak ada hasil yang ditemukan.")

except Exception as e:

print(f"Error executing query: {e}")

messagebox.showerror("Error", "Terjadi kesalahan saat menjalankan pencarian.")

finally:

conn.close()

def update\_total\_expense():

try:

conn = sqlite3.connect('fintrack.db')

cursor = conn.cursor()

cursor.execute("SELECT SUM(amount) FROM financial\_record WHERE type = 'expense' AND account\_id = ?", (self.current\_user\_id,))

result = cursor.fetchone()

total\_expense = result[0] if result[0] else 0

conf\_label\_RP1.configure(text=f"Rp. {total\_expense:,}")

conn.close()

except Exception as e:

messagebox.showerror("Database Error", f"An error occurred: {e}")

def update\_total\_income():

try:

conn = sqlite3.connect('fintrack.db')

cursor = conn.cursor()

cursor.execute("SELECT SUM(amount) FROM financial\_record WHERE type = 'income' AND account\_id = ?", (self.current\_user\_id,))

result = cursor.fetchone()

total\_income = result[0] if result[0] else 0

conf\_label\_RP.configure(text=f"Rp. {total\_income:,}")

conn.close()

except Exception as e:

messagebox.showerror("Database Error", f"An error occurred: {e}")

def delete():

selected\_item = self.tree.focus()

if not selected\_item:

messagebox.showerror("Error", "No item selected to delete!")

return

row = self.tree.item(selected\_item)['values']

confirm = messagebox.askyesno("Confirm", "Are you sure you want to delete this record?")

if not confirm:

return

try:

conn = sqlite3.connect('fintrack.db')

cursor = conn.cursor()

cursor.execute("DELETE FROM financial\_record WHERE date = ? AND description = ? AND amount = ? AND type = ?", (row[0], row[1], row[2], row[3]))

conn.commit()

conn.close()

self.tree.delete(selected\_item)

clear()

load\_data()

first\_conditional\_button()

messagebox.showinfo("Success", "Record deleted successfully!")

except Exception as e:

messagebox.showerror("Error", f"An error occurred: {e}")

def edit():

selected\_item = self.tree.focus()

if not selected\_item:

messagebox.showerror("Error", "No item selected to edit!")

return

row = self.tree.item(selected\_item)['values']

date = combobox\_CRUD\_date.get()

amount = entry\_CRUD\_nominal.get()

description = entry\_CRUD\_description.get()

record\_type = selected\_choice.get()

if not (date and description and amount and record\_type):

messagebox.showerror("Error", "Please enter all fields!")

return

if not amount.isdigit():

messagebox.showerror("Error", "Amount must be numeric!")

return

try:

conn = sqlite3.connect('fintrack.db')

cursor = conn.cursor()

cursor.execute('''UPDATE financial\_record

SET date = ?, description = ?, amount = ?, type = ?

WHERE account\_id = ? AND date = ? AND description = ? AND amount = ? AND type = ?''',(date, description, amount, record\_type, self.current\_user\_id, row[0], row[1], row[2], row[3]))

conn.commit()

conn.close()

self.tree.item(selected\_item, values=(date, description, amount, record\_type))

clear()

first\_conditional\_button()

load\_data()

messagebox.showinfo("Success", "Record updated successfully!")

except Exception as e:

messagebox.showerror("Error", f"An error occurred: {e}")

def display\_data(event=None):

selected\_item = self.tree.focus()

if selected\_item:

second\_conditional\_button()

row = self.tree.item(selected\_item)['values']

clear()

combobox\_CRUD\_date.insert(0, row[0])

entry\_CRUD\_nominal.insert(0, row[2])

entry\_CRUD\_description.insert(0, row[1])

selected\_choice.set(row[3])

else:

pass

def clear(clicked=None):

if clicked:

first\_conditional\_button()

self.tree.selection\_remove(self.tree.focus())

self.tree.focus('')

combobox\_CRUD\_date.delete(0, tk.END)

entry\_CRUD\_nominal.delete(0, tk.END)

entry\_CRUD\_description.delete(0, tk.END)

selected\_choice.set(None)

def load\_data():

conn = sqlite3.connect("fintrack.db")

cursor = conn.cursor()

cursor.execute("SELECT date, description, amount, type FROM financial\_record WHERE account\_id = ?", (self.current\_user\_id,))

rows = cursor.fetchall()

conn.close()

for item in self.tree.get\_children():

self.tree.delete(item)

for row in rows:

self.tree.insert("", 0, values=row)

update\_total\_income()

update\_total\_expense()

def insert():

date = combobox\_CRUD\_date.get()

amount = entry\_CRUD\_nominal.get()

description = entry\_CRUD\_description.get()

type = selected\_choice.get()

if not (date and description and amount and type):

messagebox.showerror('Error', 'Please enter all fields!')

return

if not amount.isdigit():

messagebox.showerror('Error', 'Amount must be a numeric value!')

return

try:

conn = sqlite3.connect('fintrack.db')

cursor = conn.cursor()

cursor.execute('INSERT INTO financial\_record (account\_id, date, description, amount, type) VALUES (?, ?, ?, ?, ?)', (self.current\_user\_id, date, description, amount, type))

conn.commit()

clear()

load\_data()

update\_total\_income()

messagebox.showinfo('Success', 'Data has been added!')

conn.close()

except Exception as e:

messagebox.showerror('Database Error', f'An error occurred: {e}')

def clear\_display():

first\_conditional\_button()

combobox\_CRUD\_date.delete(0, 'end')

entry\_CRUD\_nominal.delete(0, 'end')

entry\_CRUD\_description.delete(0, 'end')

selected\_choice.set(None)

for selected\_item in self.tree.selection():

self.tree.selection\_remove(selected\_item)

load\_data()

def update\_time():

current\_time = time.strftime("%H:%M:%S")

label\_clock.configure(text=current\_time)

label\_clock.after(1000, update\_time)

def second\_conditional\_button():

self.button\_delete\_CRUD.configure(state='normal')

self.button\_edit\_CRUD.configure(state='normal')

self.button\_add\_CRUD.configure(state='disabled')

def first\_conditional\_button():

self.button\_delete\_CRUD.configure(state='disabled')

self.button\_edit\_CRUD.configure(state='disabled')

self.button\_add\_CRUD.configure(state='normal')

def next\_patch\_page():

for widget in self.window.winfo\_children():

widget.destroy()

self.next\_patch()

# Frame Background

frame\_bg = ctk.CTkFrame(self.window, fg\_color='#c1ff72')

frame\_bg.place(x=0, y=0, relwidth=1, relheight=1)

# Frame widget Sidebar

frame\_sidebar = ctk.CTkFrame(frame\_bg,fg\_color="#00712d", width=180, border\_width=3,border\_color="#33372C")

label\_sidebar = ctk.CTkLabel(frame\_sidebar,image=self.po, text="", )

button1\_sidebar = ctk.CTkButton(frame\_sidebar, text='Dashboard', fg\_color='#264742', width=120, height=35, corner\_radius=0, border\_width=0, border\_color= '#711DB0', state='disabled')

button2\_sidebar = ctk.CTkButton(frame\_sidebar, text='Comingsoon', fg\_color='#7ed957',width=120, height=35,corner\_radius=0,border\_width=0, command= next\_patch\_page)

button\_logout = ctk.CTkButton(frame\_sidebar,text='',image=self.logout,width=40, height=40,bg\_color="#00712d",fg\_color="#00712d", command=logout)

frame\_sidebar.pack(padx=5, pady=10, side="left", fill="y")

label\_sidebar.pack(padx= 2.5, pady=(10,0), side="top", expand=0)

button1\_sidebar.pack(pady=(10,0), side="top", fill='x')

button2\_sidebar.pack(side="top", fill='x')

button\_logout.pack(pady= 5,side='bottom', expand=0)

# Frame Input

frame\_input = ctk.CTkFrame(frame\_bg, fg\_color="#c1ff72", width=240)

frame\_title = ctk.CTkFrame(frame\_input, width=240, height=245,fg\_color='#c1ff72')

frame\_title\_clock = ctk.CTkFrame(frame\_title,fg\_color="#eed1b0",width=240,height=50, corner\_radius=5, border\_width=3, border\_color='#33372C')

frame\_finder = ctk.CTkFrame(frame\_title, fg\_color="#ffde59", width=240, height=185, corner\_radius=5, border\_width=3 , border\_color='#33372C')

frame\_CRUD= ctk.CTkFrame(frame\_input, width=230,height=0, fg\_color='#d1c48f', corner\_radius=3, border\_width=3, border\_color='#33372C')

frame\_input.pack(padx=(8,10), pady=(0,10), side="right", fill="y")

frame\_input.pack\_propagate(False)

frame\_title.pack(side='top',pady=10)

frame\_title.pack\_propagate(False)

frame\_title\_clock.pack(side='top')

frame\_title\_clock.pack\_propagate(False)

frame\_finder.place(x=0, y=60)

frame\_CRUD.pack(pady=(5,0), side='bottom',fill='y', expand=1 )

label\_clock= ctk.CTkLabel(frame\_title\_clock, text='test', font=('arial', 20, 'bold'), corner\_radius=15)

label\_clock.pack(side="top",pady=10)

update\_time()

label\_title\_quote= ctk.CTkLabel(frame\_title, text="Quote", font=("Arial", 36,'bold'),fg\_color='#ffde59')

label\_title\_quote.place(relx=0.5, rely=0.35, anchor="center")

label\_quote=ctk.CTkLabel(frame\_finder, text=random.choice(self.quotes), font=("Arial", 15, "normal"), wraplength=200 )

label\_quote.place(relx=0.1,rely=0.1, relwidth=0.8,relheight=0.8)

# CRUD

frame\_CRUD.columnconfigure((0,1,2,3,4,5,6), weight=1)

frame\_CRUD.rowconfigure((0,1,2,3,4,5,6,7,8,9,10),weight=1)

selected\_choice = tk.StringVar()

label\_data\_entry = ctk.CTkLabel(frame\_CRUD, text="Data Entry", font=("Arial", 20,'bold'))

label\_CRUD\_date= ctk.CTkLabel(frame\_CRUD, text="Date", font=("Arial", 15))

combobox\_CRUD\_date = DateEntry(frame\_CRUD, date\_pattern="yyyy-mm-dd", background='darkblue', foreground='yellow', selectbackground="blue", fieldbackground='#d9d9d9')

label\_CRUD\_nominal = ctk.CTkLabel(frame\_CRUD, text= "Nominal :")

entry\_CRUD\_nominal = ctk.CTkEntry(frame\_CRUD, width=100)

label\_CRUD\_description = ctk.CTkLabel(frame\_CRUD, text= "Description:")

entry\_CRUD\_description = ctk.CTkEntry(frame\_CRUD, width=100)

label\_type = ctk.CTkLabel(frame\_CRUD, text="Type")

radiobutton\_CRUD1 = ctk.CTkRadioButton(frame\_CRUD,text="Income", value="income",variable=selected\_choice)

radiobutton\_CRUD2 = ctk.CTkRadioButton(frame\_CRUD, text="Expense", value="expense", variable=selected\_choice )

self.clear\_button= ctk.CTkButton(frame\_CRUD, text="CLEAR",corner\_radius=0,fg\_color="red",command=clear\_display, text\_color="black")

self.button\_add\_CRUD = ctk.CTkButton(frame\_CRUD, text="Add",command= insert ,corner\_radius=0, fg\_color="white", text\_color='black')

self.button\_edit\_CRUD = ctk.CTkButton(frame\_CRUD, text="Update", command= edit,corner\_radius=0, fg\_color="white", text\_color='black',)

self.button\_delete\_CRUD = ctk.CTkButton(frame\_CRUD, text="Delete", command = delete,corner\_radius=0, fg\_color="white", text\_color='black', )

button\_find\_CRUD = ctk.CTkButton(frame\_CRUD, text="Find", command = search\_transactions,corner\_radius=0, fg\_color="pink", text\_color='black')

#layout finder, progressbar, CRUD

label\_data\_entry.grid(padx=3,pady=(7,0), row=0, column=0, columnspan=7,rowspan=2, sticky="nsew")

label\_CRUD\_date.grid(padx=3,row=2, column=0,columnspan=3, )

combobox\_CRUD\_date.grid(padx=3, row=3, column=0,columnspan=3)

label\_CRUD\_nominal.grid(row=4,column=0, columnspan=3)

entry\_CRUD\_nominal.grid(padx=3,row=5,column=0, columnspan=3)

label\_CRUD\_description.grid(row=6,column=0,columnspan=3)

entry\_CRUD\_description.grid(padx=3,row=7,column=0, columnspan=3)

label\_type.grid(row=3, column=3, columnspan=3, rowspan=2)

radiobutton\_CRUD1.grid(row=4, column=3, columnspan=4, rowspan=2)

radiobutton\_CRUD2.grid(row=5, column=3, columnspan=4,rowspan=2)

self.clear\_button.grid(padx=(0,3),row=8, column=5, columnspan=2)

self.button\_add\_CRUD.grid(padx= (3,0),row=9, column=0,columnspan=4)

self.button\_edit\_CRUD.grid(padx=(0,3),pady=(0,7),row=10, column=4, columnspan=3)

self.button\_delete\_CRUD.grid(padx= (3,0),pady=(0,7),row=10, column=0, columnspan=4, )

button\_find\_CRUD.grid(padx=(0,3),row=9, column=4, columnspan=3)

# Frame Header Bar

frame\_headbar = ctk.CTkFrame(frame\_bg, height=60, fg\_color="#5DB996", border\_width=3, border\_color='#33372C')

label\_dashboard = ctk.CTkLabel(frame\_headbar, image=self.photo, text='')

frame\_headbar.pack(pady=(10,5), side="top", fill='x')

frame\_headbar.pack\_propagate(False)

label\_dashboard.place(relx=0.5,rely=0.5, relheight= 0.90, relwidth=0.96, anchor='center')

# Frame Content 1

frame\_content = ctk.CTkFrame(frame\_bg, width =598 , height=170, fg\_color="blue", border\_width=3, border\_color='#33372C')

label\_hello = ctk.CTkLabel(frame\_content, text="", image= self.hello,)

label\_username = ctk.CTkLabel(frame\_content, text=self.logged\_in\_username, font=self.font3, fg\_color='#5eaab2')

frame\_content.pack(pady=(0, 5), side='top', fill=None, expand=0)

frame\_content.pack\_propagate(False)

frame\_content.pack(pady=(0, 5), side='top', fill=None, expand=0)

frame\_content.pack\_propagate(False)

label\_hello.place(relx=0.5,rely=0.4975, relwidth=0.99 ,relheight=0.97, anchor='center')

label\_username.place(x=17, y=85, )

# frame widget Content 2 (Subframes)

frame\_content1 = ctk.CTkFrame(frame\_bg, fg\_color="#c1ff72", width=596, height=140)

frame\_a = ctk.CTkFrame(frame\_content1, fg\_color="purple", width=296, height=140, corner\_radius=3)

label\_income= ctk.CTkLabel(frame\_a,text='', image=self.income)

frame\_b = ctk.CTkFrame(frame\_content1, fg\_color="purple", width=296, height=140,)

label\_expense= ctk.CTkLabel(frame\_b,text='', image=self.expense)

# layout frame widget content 2

frame\_content1.pack(pady=(0,5), side='top', expand=0)

frame\_content1.pack\_propagate(False)

frame\_a.place(x=0,y=0)

frame\_a.pack\_propagate(False)

label\_income.place(x=0,y=0)

frame\_b.place(x=301, y=0)

frame\_b.pack\_propagate(False)

label\_expense.place(x=0,y=0)

# widget content 2 a

conf\_label\_RP = ctk.CTkLabel(frame\_a, text='Rp. 0' , font= self.font4, fg\_color = '#06d001')

conf\_label\_RP.place(x=14,y=61, anchor= 'w')

# widget content 2 b

conf\_label\_RP1 = ctk.CTkLabel(frame\_b, text='Rp. 0', font= self.font4, fg\_color='#fb4141')

conf\_label\_RP1.place(x=14,y=61, anchor= 'w')

# Frame Content 3

frame\_content2 = ctk.CTkFrame(frame\_bg, height=170, fg\_color="red")

frame\_content2.pack(pady=(0,5), side='top', fill='x', expand=0)

frame\_content2.pack\_propagate(False)

style= tk.ttk.Style(self.window)

style.theme\_use('clam')

style.configure('Treeview',font=('arial', 10),foreground='black',background='#ffde59', fieldbackground='#d9d9d9',borderwidth=0)

style.configure('Treeview.Heading', background = '#B1F0F7')

style.map('Treeview', background=[('selected', '#685752')])

self.tree = tk.ttk.Treeview(frame\_content2, columns=("DATE", "DESCRIPTION", "AMOUNT", "TYPE"), show="headings")

self.tree.column('DATE', anchor=tk.CENTER, width=80)

self.tree.column('DESCRIPTION', anchor=tk.CENTER, width=80)

self.tree.column('AMOUNT', anchor=tk.CENTER, width=80)

self.tree.column('TYPE', anchor=tk.CENTER, width=80)

self.tree.heading("DATE", text="Date")

self.tree.heading("DESCRIPTION", text="Description")

self.tree.heading("AMOUNT", text="Amount")

self.tree.heading("TYPE", text="Type")

self.tree.pack(fill='both', expand=True)

self.tree.bind('<ButtonRelease>', lambda event : display\_data(event))

load\_data()

first\_conditional\_button()

combobox\_CRUD\_date.delete(0, tk.END)

Secara keseluruhan, metode main\_page() yang saya buat bertanggung jawab untuk menyediakan tampilan yang informatif dan interaktif yang memungkinkan pengguna untuk memanage catatan keuangan mereka dengan mudah dan efisien.

Berikut penjelasan singkat fungsi-fungsi yang ada di dalam kode tersebut:

**1. logout():**

* Fungsi ini digunakan untuk keluar dari aplikasi. Ini akan menghapus informasi pengguna yang sedang login, membersihkan widget di halaman utama, dan menampilkan halaman login kembali. Sebuah message box menunjukkan bahwa pengguna telah berhasil logout.

**2. search\_transactions():**

* Fungsi ini memungkinkan pengguna untuk mencari transaksi berdasarkan tanggal. Fungsi ini memeriksa apakah tanggal diinputkan, lalu menampilkan data transaksi yang sesuai (berdasarkan akun pengguna yang sedang login) di dalam treeview. Jika tidak ada data, maka akan ada pemberitahuan.

**3. update\_total\_expense():**

* Fungsi ini menghitung total pengeluaran (dengan tipe 'expense') dari transaksi pengguna yang sedang login dan menampilkannya pada label yang menunjukkan total pengeluaran di layar.

**4. update\_total\_income():**

* Fungsi ini menghitung total pendapatan (dengan tipe 'income') dari transaksi pengguna yang sedang login dan menampilkannya pada label yang menunjukkan total pendapatan di layar.

**5. delete():**

* Fungsi ini memungkinkan pengguna untuk menghapus transaksi yang dipilih dari treeview. Pengguna diminta untuk mengonfirmasi penghapusan, dan jika dikonfirmasi, data transaksi tersebut akan dihapus dari database.

**6. edit():**

* Fungsi ini memungkinkan pengguna untuk mengedit transaksi yang dipilih dari treeview. Setelah memilih transaksi, pengguna dapat mengubah informasi seperti tanggal, deskripsi, jumlah uang, dan tipe transaksi. Fungsi ini kemudian memperbarui database dengan perubahan tersebut.

**7. display\_data(event=None):**

* Fungsi ini dipanggil ketika sebuah baris pada treeview dipilih. Fungsi ini mengisi input field (seperti tanggal, deskripsi, jumlah, dan tipe) dengan data dari transaksi yang dipilih.

**8. clear(clicked=None):**

* Fungsi ini menghapus semua nilai yang ada pada input fields (seperti tanggal, jumlah, deskripsi, tipe), serta mengatur ulang status tombol (seperti tombol Edit dan Delete).

**9. load\_data():**

* Fungsi ini mengambil semua data transaksi dari database yang terkait dengan akun pengguna yang sedang login dan menampilkannya di treeview. Selain itu, ini juga memperbarui total pemasukan dan pengeluaran.

**10. insert():**

* Fungsi ini menambahkan data transaksi baru ke database. Pengguna harus mengisi semua field (tanggal, deskripsi, jumlah, dan tipe transaksi). Jika data valid, transaksi baru akan disimpan di database.

**11. clear\_display():**

* Fungsi ini membersihkan form input dan mengatur ulang tampilan kembali ke kondisi awal. Ini juga memuat ulang data transaksi yang ada.

**12. update\_time():**

* Fungsi ini memperbarui label jam setiap detik untuk menampilkan waktu saat ini.

**13. Conditional Buttons:**

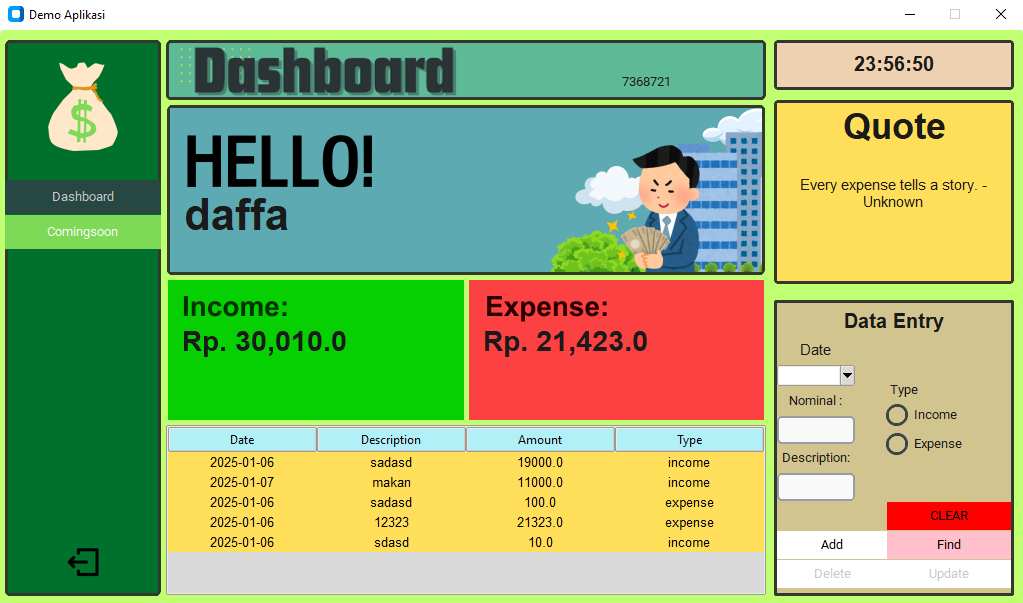
* **second\_conditional\_button()**: Fungsi ini mengaktifkan tombol Edit dan Delete, serta menonaktifkan tombol Add ketika ada transaksi yang dipilih di treeview.
* **first\_conditional\_button()**: Fungsi ini mengaktifkan tombol Add dan menonaktifkan tombol Edit dan Delete ketika tidak ada transaksi yang dipilih.

**14. Treeview (Data Display):**

* **self.tree** adalah widget **treeview** yang menampilkan data transaksi dalam bentuk tabel dengan kolom untuk tanggal, deskripsi, jumlah, dan tipe transaksi. Data di sini dimuat menggunakan fungsi load\_data() dan bisa diedit atau dihapus oleh pengguna.

**15. next\_patch\_page():**

* Fungsi ini digunakan untuk beralih ke halaman selanjutnya di aplikasi (dalam hal ini, halaman yang belum selesai dan diberi label "Comingsoon").

Untuk sisa kode lainnya berfungsi untuk menampilkan dan penataan elemen-elemen antarmuka pengguna (UI), seperti frame, tombol, label, dan treeview. Misalnya, frame sidebar yang berisi tombol navigasi, input form untuk memasukkan data, serta treeview untuk menampilkan data transaksi. termasuk pengaturan layout antarmuka pengguna. Sidebar berisi tombol untuk navigasi.

def next\_patch(self):

def dashboard\_page():

for widget in self.window.winfo\_children():

widget.destroy()

self.main\_page()

# Frame Background

frame\_bg = ctk.CTkFrame(self.window, fg\_color='#c1ff72')

frame\_bg.place(x=0, y=0, relwidth=1, relheight=1)

#label

label = ctk.CTkLabel(frame\_bg, text='Stay Tune...', font= ("arial", 30, 'bold'))

label.place(relx=0.5, rely=0.5)

# Frame widget Sidebar

frame\_sidebar = ctk.CTkFrame(frame\_bg,fg\_color="#00712d", width=180, border\_width=3,border\_color="#33372C")

label\_sidebar = ctk.CTkLabel(frame\_sidebar,image=self.po, text="", )

button1\_sidebar = ctk.CTkButton(frame\_sidebar, text='Dashboard', fg\_color='#7ed957' ,width=120, height=35, corner\_radius=0, border\_width=0, border\_color= '#711DB0', command=dashboard\_page)

button2\_sidebar = ctk.CTkButton(frame\_sidebar, text='Comingsoon', fg\_color='#264742',width=120, height=35,corner\_radius=0,border\_width=0, state='disabled')

button\_logout = ctk.CTkButton(frame\_sidebar,text='',image=self.logout,width=40, height=40,bg\_color="#00712d",fg\_color="#00712d", command=self.logout)

frame\_sidebar.pack(padx=5, pady=10, side="left", fill="y")

label\_sidebar.pack(padx= 2.5, pady=(10,0), side="top", expand=0)

button1\_sidebar.pack(pady=(10,0), side="top", fill='x')

button2\_sidebar.pack(side="top", fill='x')

button\_logout.pack(pady= 5,side='bottom', expand=0)

def run(self):

self.window.mainloop()

Secara umum pada metode next\_patch(): tidak ada fungsi berarti dikarenakan ini hanyalah halaman kosong, dalam halaman ini masih pada tahap pengerjaan.

Dan Metode run(): memiliki fungsi menjalankan aplikasi dengan memulai mainloop(), yang menjaga antarmuka tetap aktif dan menunggu interaksi pengguna.



if \_\_name\_\_ == "\_\_main\_\_":

app = App()

app.run()

if \_\_name\_\_ == "\_\_main\_\_": digunakan untuk memastikan bahwa bagian kode di bawahnya hanya dijalankan jika file ini dijalankan langsung sebagai program utama, bukan ketika file ini diimpor ke file Python lain. “app = App()” menjadikan kelas utama sebagai objek dan “app.run()” berfungsi untuk menjalankan metode run() yang ada di dalam kelas App, yang dapat untuk menjalankan aplikasi